LAB 1: MetaMask, Ethereum and Blocks

1. Introduction

In our lectures, we have explicitly introduced blockchain and cryptography. In order to give you a deep understanding on how blocks work in real-world application, this lab will show the logic behind building blocks by using MetaMask wallet and creating Ethereum transactions. If you want to find a more comprehensive guide for MetaMask, you can check MetaMask Docs at https://docs.metamask.io/guide/#why-metamask.

By creating Ethereum accounts and making several transactions in MetaMask, we will have an in-depth look at some properties of Ethereum transaction and Cryptographic Hashing in order to fully understand the authenticity and security of Ethereum transactions.

2. MetaMask

2.1 Introduction

MetaMask is a plug-in Ethereum crypto wallet for Chrome onboard users. Available as a browser extension and as a mobile app, MetaMask equips us with a key vault, secure login, and token wallet—everything we need to manage our digital assets. MetaMask provides the simplest yet most secure way to connect to blockchain-based applications. In this and following labs, we will use MetaMask to store and send tokens not only between our accounts, but also in smart contracts which will be covered in the following lectures and labs.

2.2 MetaMask Setup

Complete information and study guide about MetaMask can be found at its official website metamask.io. We need to choose the right browser (Chrome is recommended) and follow its installation instruction. When we are creating a new MetaMask account, here are some key points we need to pay attention to.

First of all, creating a new strong password is extremely important because it encrypts private key. As we discussed in the lecture, private keys give access to all of our Ether or other tokens. So, it is better to have a strong password here.

Secret Backup Phrase, which includes 12 mnemonic words, will pop out after setting up the password. We need to write this phrase on a piece of paper or store it in a secure location because secret backup phrase makes easier to back up and restore our account if we log out our account or accidentally clear browser history.

We are now able to use interact with MetaMask.

2.3 Deposit Ether

Following steps can be completed on either MetaMask website or its extension interface (we can enter the interface from browser's extension toolbar, which is on the top-left corner for Chrome). First of all, we have to choose a right network to make our first transaction. There are several options for the networks: Ethereum Mainnet, local host, and test networks, which include Rinkeby, Goerli test network, Sepolia test network. The default network is Main Ethereum Network where we can trade real Ether. In this and following labs, we are going to use Sepolia test network (you may also use other test networks as well) in which we can use free test Ether from the third-party websites. Testing on Main Network may cost too much real Ether which is unaffordable for developer like us. By clicking "Ethereum Mainnet" in website or interface, it will give us several choices and we are going to choose "Sepolia test network" for our developers' environment.

TO DO 1:
Deposit some Ether in your MetaMask accounts.

You need to copy Sepolia test network account address from MetaMask and enter it into https://sepolia-faucet.pk910.de/. The website simulates the real mining process that you can get Sepolia test tokens if you spend time and computational power to mine the block. The mining speed is around 5 Sepolia Ether per hour.

2.4 Make A Transaction

In this section, we are going to make a transaction between our accounts. We only have one default account right now, but we definitely need to create more accounts.

By clicking the top-right account picture in MetaMask interface, we can see 'Create Account' button. By clicking that button and entering the account name, we will switch back to the account where we deposited Ether, click 'Send', enter an amount of Ether, select an account we want to transfer to and choose the speed to send Ether. Depending on the speed we choose, the transaction usually takes about 15 - 30 seconds.

While we are waiting for it to be completed, we can find this transaction in 'Queue' (or in 'History' if the transaction is finished). By clicking 'View on Etherscan', we will find transaction details including Transaction Hash, Status, Block, Timestamp, From, To, Transaction Value and Transaction Fee. As we can see, our transaction has been recorded in Rinkeby Network, and anybody in the Rinkeby can see our blocks.

TO DO 2:
Create several accounts and make some transactions between these accounts.

3. Cryptographic Hashing

3.1 Introduction

Cryptographic Hashing function plays an important role in blockchain and any digital currencies related with that. In this section, we will talk about how cryptographic hash works; however, discussion about its properties is necessary. By analyzing and understanding these properties, we will know how blocks are chained together and how Ethereum or other blockchain networks operate.

3.2 Properties of Cryptographic Hashing

Because we have discussed these properties in our lectures, there will be no detailed explanation here. These properties are:

1) Deterministic
2) Quick to compute
3) Impossible to go back
4) Small change in input results in big change in output
5) Different messages generate different hash values

TO DO 3:
Test some properties of cryptographic hashing in Section 3.2.

The first website below has two fields: the upper one is 'Data' and the lower one is 'Hash'. You are going to enter some value in 'Data' field and check 'Hash' value in order to test some of properties. If you are interested in mining block, blockchain, distributed blockchain, tokens or coinbase transactions, please watch the tutorial video in the second link and practice them in the links starting from the third one.

https://andersbrownworth.com/blockchain/hash
https://www.youtube.com/watch?time_continue=5&v=_160oMzblY8&feature=emb_logo
https://andersbrownworth.com/blockchain/block
https://andersbrownworth.com/blockchain/blockchain
https://andersbrownworth.com/blockchain/distributed
https://andersbrownworth.com/blockchain/tokens
https://andersbrownworth.com/blockchain/coinbase

4. Ethereum Transaction

4.1 Introduction

In this section, we are going to have more in-depth look at how Ethereum transactions work and how to make sure that these transactions are authentic. By analyzing and understanding a classical example of an Ethereum transaction, we will fully understand the composition and authenticity of Ethereum transaction.

4.2 Preparing for Transaction Programmatically

If we look at any library which can be used to programmatically send a transaction to the network, like Web3.js library which we are going to work with later, we will see that a transaction object contains several parameters. Some of them are required and some of them are optional. Let's have a closer look at a classical transaction object which just like the transaction we sent in TO DO 2:

1) from: the account where we send Ether from;
2) to: (optional) the account where we send Ether to;
3) value: (optional) amount of Ether we send in Wei (1 Ether = 10^18 Wei);
4) gas: (optional) maximum amount of gas in a transaction;
5) gasPrice: (optional) amount that sender pays per computational step;
6) data: (optional) ABI byte strings.
7) nonce: (optional) integer of a nonce

First three items are relatively easy to understand. Gas is a special unit in Ethereum and it refers to the cost necessary to perform a transaction. Ethereum gas functions similarly as gasoline in the car. Gas limit determines how much gas we can use in a transaction, and if we used up all gas, the entire transaction will be terminated. Gas limit works similarly with fuel tank, the volume of fuel tank cannot be changed once a car is produced and so does gas limit which cannot be changed once the transaction starts. MetaMask wallet will automatically set gas limit as 21,000 units when we have some simple transfers, but when the transaction becomes more complex or requires more computational steps, gas limit should accordingly go up. Depending on different computational steps and functions that we call, gas prices for transaction vary from zero to a couple of thousands Gwei (1 Gwei = 10^9 Wei). If you want to find specific gas price, please check Appendix G. Fee Schedule at:

https://ethereum.github.io/yellowpaper/paper.pdf

Data is an ABI (Application Binary Interface) byte string using in Smart Contracts when they are deployed to the blockchain. And we will discuss about data in detail when we start to design Smart Contract. Nonce is an integer which is incremented every time a transaction is sent in order to avoid replay attacks.

4.3 Ethereum Transaction Signature

We have learnt Cryptographic Hashing Functions and Digital Signature with Elliptic Curve in our lectures. We also discussed User Identity (Private Key and Public Key). In this section, we are going to expand our views by analyzing the authenticity of Ethereum transaction. Following equations play important roles in producing authentic transaction:

1) A Transaction Object + Private Key => Signed Transaction
2) Private Key + Elliptic Curve Digital Signature Algorithm (ECDSA) => Public Key
3) Public Key + Keccak Hash=> Ethereum Account
4) Signed Transaction + ECRECOVER => Ethereum Account

As we discussed in Section 2, a transaction object contains different parameters. Elliptic Curve Digital Signature Algorithm (ECDSA) ensures that, besides the owner of Private Key, anyone cannot generate the owner's Private Key from his/her Ethereum Account. If you are interested in ECDSA, please find more information at:

https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages

Keccak Hash is best known as hash function. It is widely used for authentication, encryption and pseudo-random number generation. For an explanation on how Ethereum uses Keccak Hash function, specifically Keccak-256, to generate Ethereum Account address, please check tayvano's reply here:

https://ethereum.stackexchange.com/questions/3542/how-are-ethereum-addresses-generated

ECRECOVER stands for Elliptic Curve Recover Function and it transforms signed transaction to Ethereum Account. If you want to get more information about how it works, please check the following website:

https://gist.github.com/axic/5b33912c6f61ae6fd96d6c4a47afde6d

We can generate Ethereum Account in two different ways which can help us verify the authenticity of an Ethereum transaction.

TO DO 4:
Add transactions to a blockchain.
For safety and security consideration, we want to add transactions to a blockchain that only the owner of private key can create these transactions. Watch the tutorial video in the first link and practice in the links starting from the second one.

https://www.youtube.com/watch?time_continue=1&v=xIDL_akeras&feature=emb_logo
https://andersbrownworth.com/blockchain/public-private-keys/keys
https://andersbrownworth.com/blockchain/public-private-keys/signatures
https://andersbrownworth.com/blockchain/public-private-keys/transaction
https://andersbrownworth.com/blockchain/public-private-keys/blockchain